МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ

ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

для студентів спеціальностей
122 – Комп'ютерні науки та
124 – Системний аналіз

Затверджено
редакційно – видавничою
радою університету,
протокол № 3 від 06.11.2019 р.

Харків
НТУ «ХПІ»
2019

Методичні вказівки до лабораторних робіт з дисципліни «Штучні нейронні мережі» для студентів спеціальностей 122 – Комп'ютерні науки та 124 – Системний аналіз / уклад. Ю.І. Дорофєєв. – Харків: НТУ «ХПІ», 2019. – 40 с.

Укладач Дорофєєв Ю. І.

Рецензент проф. Любчик Л. М.

Кафедра системного аналізу та інформаційно-аналітичних технологій

Вступ

Метою виконання лабораторних робіт є знайомство з різними типами архітектур штучних нейронних мереж, способами їх побудови, навчання та моделювання. Штучні нейронні мережі (ШНМ) – це окремий напрямок розвитку інтелектуальних систем, які знайшли застосування в самих різних галузях людської діяльності. Можливості нейронних мереж дозволяють застосовувати їх для вирішення завдань розпізнавання образів, кластеризації даних, асоціативного (контекстного) пошуку, апроксимації функції багатьох змінних, прогнозування, оптимізації та інших завдань, які відносяться до класу тих, що важко формалізуються.

Для проектування та моделювання ШНМ застосовуються спеціалізовані програмні пакети, одним з яких є програмний комплекс MATLAB – матрична лабораторія, призначена для різноманітних технічних обчислень. До складу MATLAB включено набір пакетів розширення (тулбоксів) для розв'язання спеціалізованих задач, що відносяться до певних областей обробки даних.

Специфічні процедури та функції, що призначені для моделювання нейронних мереж, об'єднані в Neural Network Toolbox. Успіхи, які в останні роки були досягнуті за допомогою ШНМ в розпізнаванні зображень та природної мови, а також значні ресурси, що в даний час вкладаються в розробку програмного забезпечення та апаратних засобів для реалізації штучних нейронних мереж, доводять, що існує велика зацікавленість в подальшому розвитку ШНМ.

У даному посібнику подано лише основні типи архітектури нейронних мереж та використовувані в них методи і процедури, а також наведено приклади їх застосування для вирішення конкретних завдань. Наявність в MATLAB широкого спектру графічних можливостей дозволяє користувачеві наочно уявляти архітектуру нейронних мереж, а також графічно відображати отримані результати моделювання.

Дані лабораторні роботи спрямовані на набуття практичних навичок використання сучасних програмних засобів для створення та моделювання штучних нейронних мереж.

ЛАБОРАТОРНА РОБОТА 1

Класифікація образів за допомогою одношарового персептрона

1.1. Інформаційний матеріал

Для розв'язання будь-якої прикладної задачі за допомогою штучної нейронної мережі необхідно виконати п'ять етапів: підготувати дані для навчання, сформувати архітектуру ШНМ, ініціалізувати мережу, виконати навчання, протестувати навчену ШНМ.

Для вирішення задач класифікації вхідних векторів, які відносяться до класу лінійно роздільних, використовуються одношарові нейронні мережі, які називають персептрони.

Формування архітектури мережі.

Для створення ШНМ, яка має архітектуру одношарового персептрона, в пакеті Neural Network Toolbox використовується функція **perceptron**:

> net = perceptron (hardlimitTF, perceptronLF).

Вхідні аргументи:

hardlimitTF – активаційна функция (за замовчуванням '**hardlim**' – порогова);

perceptronLF – функція навчання (за замовчуванням '**learnp**').

Вихідні аргументи:

net – об'єкт класу Neural Network з архітектурою одношарового персептрона. Кількість нейронів персептрона визначається автоматично в процесі навчання, виходячи з того, що одношаровий персептрон, який містить m нейронів, здатен розділити вхідний простір на 2^m класів. Значення ваг та зміщень персептрона за замовчуванням ініціалізуються за допомогою функції **init**, яка встановлює нульові значення.

Існує інший спосіб сформувати одношаровий персептрон за допомогою функції **newp**:

> net = newp (X, T, s, hardlimitTF, perceptronLF).

Вхідні аргументи:

X – масив розмірністю R×Q, який містить вхідні вектори, де R – кількість елементів, з яких складений вхідний вектор, Q – кількість векторів;

T – масив розмірністю S×Q, який містить вихідні вектори, де S – кількість елементів, з яких складений вихідний вектор;

s – кількість нейронів.

Приклад. Сформувати персептрон для розподілу трьох 2-елементних векторів [0; 0], [1; 1] та [2; 2] на два класи: 0 та 1:

> X = [0 1 2; 0 1 2];

> T = [0 1 1];

> net = perceptron; % або

> net = newp (X, T, 'hardlim', 'learnp').

Ініціалізація мережі.

Після того як сформована архітектура мережі, необхідно задати початкові значення ваг та зміщень. Для цього можна використовувати:

а) функцію **init,** яка присвоює вагам та зміщенням нейронів персептрона нульові значення:

> net = init (net);

б) функцію **rands,** яка генерує випадкові значення масивів ваг та зміщень:

> net.IW{1, 1} = rands (s, R) – генерує матрицю розмірністю s×R, елементи якої приймають випадкові значення з інтервала [–1; 1], та присвоює отримані значення матриці ваг нейронів першого шару персептрона;

> net.b{1} = rands (s) – генерує вектор розмірністю s×1, елементи якого приймають випадкові значення з інтервала [–1; 1], та присвоює отримані значення вектору зміщень нейронів першого шару персептрона;

в) встановити бажані значення ваг та зміщень.

Приклад. Присвоїти вагам та зміщенню одношарового персептрона, який містить один нейрон, наступні значення: $w_1 = -3$, $w_2 = 2$, $w_0 = 5$.

> net.IW{1, 1} = [–3; 2];

> net.b{1} = 5.

Підготовка даних для навчання.

Вхідні вектори X можуть мати два формати: **cell array** та **double array**. Формат **cell array** відповідає *послідовному* способу подачі даних на вхід мережі, який можна інтерпретувати в такий спосіб. На вхід мережі подається послідовність з *n* вхідних сигналів, тобто на першому кроці – перший, на другому – наступний, і так далі. У відповідь мережа генерує послідовність виходів, яка також містить *n* елементів. Результат був би той самий, якби було створено *n* однакових мереж, і на кожну мережу було подано один із вхідних векторів.

Подання вхідних даних як числового масиву в форматі **double** відповідає *пакетному* способу представлення даних, в результаті чого на виході мережі генерується відповідний масив.

Приклад. Сформувати чотири види вхідних даних: один вектор-стовпець з двох елементів; група (пакет) з двох трьохелементних векторів; група з трьох двоелементних векторів; послідовність з трьох двоелементних векторів:

1) > X1 = [1; 2];

2) > X2 = [1 4; 2 5; 3 6];

3) > X3 = [1 3 5; 2 4 6];

4) > X4 = {[1; 2] [3; 4] [5; 6]}.

Навчання мережі.

Щоб підготувати нейронну мережу для вирішення конкретної задачі, необхідно налаштувати її параметри, тобто виконати навчання. У пакеті програм Neural Network Toolbox реалізовано два способи навчання: *послідовний* та *пакетний* (груповий).

Для організації послідовного навчання використовується функція **adapt**, яка коригує значення ваг та зміщень за результатами одноразової подачі на вхід мережі масиву вхідних даних:

> net_adapt = adapt (net, X, T).

Вхідні аргументи:

net – ім'я нейронної мережі до навчання;

X – масив вхідних векторів;

T – масив бажаних значень вихідних векторів (масив цілей).

Вихідні аргументи:

net_adapt – ім'я нейронної мережі після навчання.

Приклад. Реалізувати процедуру навчання нейронної мережі, використовуючи наступну послідовність навчаючих шаблонів

$$\left\{X_1=\begin{bmatrix}-1\\-1\end{bmatrix}, T_1=0\right\}, \left\{X_2=\begin{bmatrix}-1\\1\end{bmatrix}, T_2=0\right\}, \left\{X_3=\begin{bmatrix}1\\-1\end{bmatrix}, T_3=1\right\}, \left\{X_4=\begin{bmatrix}1\\1\end{bmatrix}, T_4=1\right\}:$$

> X = {[–1; –1] [–1; 1] [1; –1] [1; 1]};

> T = {0 0 1 1};

> net_1 = adapt (net, X, T).

Необхідно зауважити, що функція **adapt** виконує одноразове коригування параметрів мережі, тобто не гарантує, що після цього мережа виконає правильну класифікацію нового вхідного вектора. Для реалізації ітеративної процедури навчання необхідно організувати цикл, на кожному кроці якого застосовується процедура **adapt** та перевіряється, чи був виконаний критерій зупинки навчання (наприклад, поточний вектор похибок мережі містить нульові значення).

Для автоматичної реалізації ітеративної процедури навчання використовується функція **train**, синтаксис якої є аналогічним. При застосуванні функції **train** у відповідне вікно на моніторі виводиться схематичне зображення побудованої мережі та в реальному часі відображаються результати навчання: кількість виконаних епох навчання та значення помилки на виході мережі (за замовчуванням – середня абсолютна помилка). Якщо задача є нерозв'язною, тоді процес навчання буде зупинений тільки після виконання встановленої кількості епох (за замовчуванням 1000). Користувач може в будь-який момент зупинити процес навчання за допомогою кнопки «Stop Training», а також вивести на монітор графік залежності величини похибки від кількості епох навчання (кнопка

«Performance»). Для зручності відображення результатів користувач може змінити логарифмічний масштаб по осі ординат (обраний за замовчуванням) на лінійний, вибравши в меню «Edit» пункт «Axes Properties».

Тестування (моделювання) мережі.

Після закінчення процесу навчання необхідно перевірити, наскільки успішно навчена ШНМ здатна вирішити конкретну задачу. Це можна зробити двома способами: за допомогою моделювання та шляхом графічного представлення результатів навчання.

Моделювання складається з подачі на вхід мережі тестових даних та порівняння результатів, які генерує мережа, з очікуваними. Для моделювання використовується функція **sim**:

> Y = sim (net_train, X).

Вхідні аргументи:

net_train – об'єкт класу Neural Network, навчання якого вже завершено;

X – масив, який містить вхідні вектори, призначені для тестування мережі.

Вихідні аргументи:

Y – масив, який містить вихідні вектори, що є результатом роботи ШНМ.

Для графічного відображення результатів навчання та моделювання використовуються функції **plotvec**, **plotpv**, **plotpc**.

Функція plotvec (X, color, Marker) відображає в двовимірному або в тривимірному просторі вектори-стовпці масиву X у вигляді маркерів, тип яких визначає строкова змінна Marker, а колір кожного маркера визначається елементами вектора-рядка color.

Приклад. Відобразити кружками різних кольорів чотири двоелементних вхідних вектора:

> X = [1.0 1.3 1.7 2.0; 0 –1.0 0.5 –1.5];

> plotvec (X, [1 2 3 4], 'o').

Функція plotpv (X, T, v) відображає в двовимірному або в тривимірному просторі вектори-стовпці масиву входів X розміру R×Q у вигляді маркерів різного типу відповідно до значень масиву цілей T розміру S×Q; вектор-рядок **v** дозволяє задати границі відображуваного простору у форматі [x_min x_max y_min y_max].

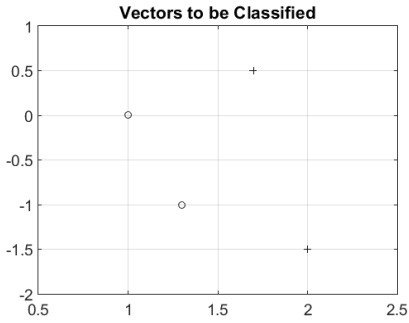
Приклад. Відобразити маркерами різного типу чотири двоелементних вектора, які містить масив X, що належать двом класам:

> T = [0 0 1 1];

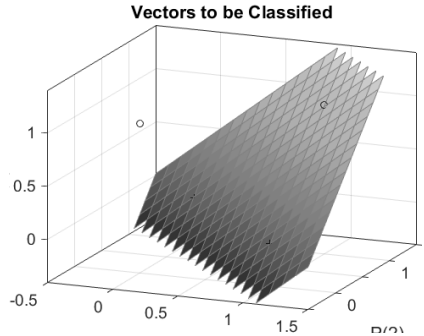
> plotpv (X, T, [0.5 2.5 –2 1]).

Результат представлено на рис. 1.1, а.

Функція plotpc (net.IW{1,1}, net.b{1}) будує лінію (в двовимірному просторі) або площину (в тривимірному), яка розділяє простір у відповідності до значень матриці ваг net.IW{1,1} та вектора зміщень net.b{1}



а



б

Рисунок 1.1

навченої нейронної мережі. Функцію plotpc доцільно використовувати разом з функцією plotpv, яка відображає розміщення в просторі вхідних векторів.

Приклад. Відобразити розміщення в тривимірному просторі чотирьох вхідних векторів, що містяться в масиві X та належать двом класам, які визначаються масивом цілей T, а також площину, відповідно до якої одношаровий персептрон, навчання якого проводилося за допомогою масивів X та T, розділяє простір на два класи:

```
> X = [0  0  1  1;   0  1  0  1;   1  0  0  1];
> T = [0  0  0  1];
> plotpv (X, T);
> hold on;
> plotpc (net.IW{1, 1}, net.b{1}).
```

Результат представлено на рис. 1.1, б.

1.2. Завдання на лабораторну роботу

1. Вибрати архітектуру та сформувати штучну нейронну мережу для вирішення задачі класифікації двоелементних образів на два класи.

Набори навчальних шаблонів представлено в таблицях:

Варіант 1

Образ	Клас
[1.0; 1.0]	1
[1.0; –1.0]	1
[2.0; 0.5]	0
[2.0; –2.0]	0

Варіант 2

Образ	Клас
[–0.3; 1.5]	0
[–2.0; 1.0]	1
[–2.0; –2.0]	1
[–0.1; –1.0]	0

Варіант 3

Образ	Клас
[–1.5; 1.5]	0
[1.5; 2.0]	0
[–0.5; 0.3]	1
[2.0; 1.0]	1

Варіант 4

Образ	Клас
[1.5; –2.0]	1
[1.0; –0.5]	0
[–1.5; –1.0]	0
[–0.5; –1.5]	1

2. Побудувати графік, який зображує простір входів, вхідні вектори та лінію, яка відображує, як навчена нейронна мережа розділяє простір входів на дві області.

3. Вибрати архітектуру та сформувати штучну нейронну мережу для вирішення задачі класифікації трьохелементних образів на два класи.

Набори навчальних шаблонів представлено в таблицях:

Варіант 1

Образ	Клас
[–3.0; –3.0; –3.0]	0
[3.0; –3.0; –3.0]	1
[–3.0; 3.0; –3.0]	1
[3.0; 3.0; –3.0]	1
[–3.0; –3.0; 3.0]	0
[3.0; –3.0; 3.0]	0
[–3.0; 3.0; 3.0]	0
[3.0; 3.0; 3.0]	1

Варіант 2

Образ	Клас
[–1.5; –1.5; –1.5]	1
[1.5; –1.5; –1.5]	0
[–1.5; 1.5; –1.5]	0
[1.5; 1.5; –1.5]	0
[–1.5; –1.5; 1.5]	1
[1.5; –1.5; 1.5]	1
[–1.5; 1.5; 1.5]	1
[1.5; 1.5; 1.5]	0

Варіант 3

Образ	Клас
[–1.0; –1.0; –1.0]	0
[1.0; –1.0; –1.0]	1
[–1.0; 1.0; –1.0]	0
[1.0; 1.0; –1.0]	0
[–1.0; –1.0; 1.0]	1
[1.0; –1.0; 1.0]	1
[–1.0; 1.0; 1.0]	0
[1.0; 1.0; 1.0]	1

Варіант 4

Образ	Клас
[–2.0; –2.0; –2.0]	1
[2.0; –2.0; –2.0]	1
[–2.0; 2.0; –2.0]	1
[2.0; 2.0; –2.0]	1
[–2.0; –2.0; 2.0]	0
[2.0; –2.0; 2.0]	0
[–2.0; 2.0; 2.0]	0
[2.0; 2.0; 2.0]	0

4. Побудувати графік, який зображує простір входів, вхідні вектори та площину, яка відображує, як навчена нейронна мережа розділяє простір входів на дві області.

5. Вибрати архітектуру та сформувати штучну нейронну мережу для вирішення задачі класифікації двохелементних образів на чотири класи.

Набори навчальних шаблонів представлено в таблицях:

Варіант 1

Образ	Клас
[0.1; 1.2]	[1; 0]
[0.7; 1.8]	[1; 0]
[0.8; 1.6]	[1; 0]
[0.8; 0.6]	[0; 0]
[1.0; 0.8]	[0; 0]
[0.3; 0.5]	[1; 1]
[0.0; 0.2]	[1; 1]
[–0.3; 0.8]	[1; 1]
[–0.5; –1.5]	[0; 1]
[–1.5; –1.3]	[0; 1]

Варіант 2

Образ	Клас
[0.5; 1.3]	[0; 1]
[0.8; 1.6]	[0; 1]
[0.9; 1.8]	[0; 1]
[1.2; 0.6]	[1; 1]
[1.5; 0.8]	[1; 1]
[0.2; 0.1]	[0; 0]
[0.1; 0.5]	[0; 0]
[–0.2; 0.6]	[0; 0]
[–0.6; –0.8]	[1; 0]
[–1.0; –1.2]	[1; 0]

Варіант 3

Образ	Клас
[–0.5; 3.0]	[1; 0]
[0.4; 2.5]	[1; 0]
[–0.8; 0.3]	[0; 0]
[–1.0; –0.4]	[0; 0]
[–0.5; –0.1]	[0; 0]
[1.3; 0.5]	[1; 1]
[1.0; –0.2]	[1; 1]
[1.5; 0.8]	[1; 1]
[–0.5; –1.5]	[0; 1]
[0.2; –1.3]	[0; 1]

Варіант 4

Образ	Клас
[0.5; 3.0]	[1; 0]
[–0.4; 2.5]	[1; 0]
[–0.8; –0.3]	[1; 1]
[–1.0; 0.4]	[1; 1]
[–0.5; 0.1]	[1; 1]
[1.3; –0.4]	[0; 0]
[1.0; 0.2]	[0; 0]
[1.5; –0.3]	[0; 0]
[0.4; –1.5]	[0; 1]
[–0.2; –1.8]	[0; 1]

6. Побудувати графік, який зображує простір входів, вхідні вектори та лінії, яка відображує, як навчена нейронна мережа розділяє простір входів на чотири області.

Контрольні питання

1. Яка активаційна функція використовується для нейронів, що утворюють персептрон?
2. Яка кількість нейронів, об’єднаних в одношаровий персептрон, знадобиться, щоб розділити вхідні образи на 10 класів?
3. Чи може одношаровий персептрон реалізувати логічну операцію XOR («виключає АБО»)? Обгрунтуйте свою відповідь.

ЛАБОРАТОРНА РОБОТА 2

Класифікація образів за допомогою персептрона, який виконує нормування вхідних даних

2.1. Інформаційний матеріал

Американський вчений Френк Розенблатт в 1957 році вперше запропонував математичну модель сприйняття інформації мозком та назвав її персептрон. Пізніше ним була доведена теорема, в якій стверджується, що елементарний персептрон, який навчається за методом корекції помилки, незалежно від початкового стану вагових коефіцієнтів та послідовності пред'явлення навчальних шаблонів завжди досягне рішення за кінцевий проміжок часу.

Однак, процедура навчання персептрона має особливості. Одна з них полягає в наступному. Якщо значення елементів деякого вхідного вектора, що використовується при навчанні, значно відрізняються одне від одного, або від значень інших векторів, то для навчання може знадобитися чимало часу. Це обумовлено тим, що алгоритм навчання пов'язаний з додаванням (або відніманням) вхідного вектора до поточного вектору ваг. Таким чином, наявність вхідного вектора з дуже великими або малими елементами призводить до того, що настройка параметрів персептрона вимагає дуже великої кількості ітерацій.

Врівноважити вплив великих чи малих компонент можна за допомогою масштабування вхідних даних. Необхідно нормувати дані так, щоб вплив будь-якого вхідного вектора мав приблизно рівний внесок. Отже, використання нормованого правила навчання персептрона значно скорочує кількість циклів навчання, коли зустрічаються викиди вхідних векторів.

Для обчислення величини корекції матриці ваг та вектора зміщень персептрона можуть застосовуватися дві функції: стандартна '**learnp**' та нормована '**learnpn**'. Нормована функція може бути вказана в якості аргументу при створенні персептрона за допомогою функції **perceptron** або **newp** (за замовчуванням використовується стандартна функція).

Крім величини норми вхідних векторів, процедура навчання персептрона є дуже чутливою до розміру навчаючої вибірки та порядку пред'явлення навчальних шаблонів. На практиці число навчальних шаблонів експоненціально зростає із зростанням кількості параметрів мережі, що настроюються. У книзі [4] наведено загальні рекомендації щодо розміру набору навчальних шаблонів. Рекомендується формувати навчальну вибірку так, щоб виконувалася наступна нерівність

$$N > \frac{W}{\varepsilon}, \tag{2.1}$$

де N – кількість навчальних шаблонів, W – кількість параметрів мережі, що настроюються, ε – відсоток помилок, допустимий в ході тестування мережі.

В процесі навчання рекомендується переміжати вхідні образи, що належать до різних класів, а також «перетасовувати» (shuffling) навчальні шаблони. Один з можливих підходів полягає в тому, щоб на кожній ітерації алгоритму випадковим чином вибирати образ навчальної вибірки (тобто під час роботи алгоритму образи навчальної вибірки перебираються в випадковому порядку, можливо, неодноразово).

Якщо застосування нормованої функції '**learnpn**' не дозволяє успішно завершити процедуру навчання персептрона, рекомендується масштабувати вхідні дані таким чином, щоб значення векторів належали інтервалу [0, 1] або [–1, 1]. Для цього необхідно знайти максимальне значення кожної з ознак $x_i^{\max}$ та розподілити значення ознаки кожного вектора на відповідні максимальні:

$$x_i^{норм} = \frac{x_i}{x_i^{\max}},\ i = \overline{1, m}, \qquad (2.2)$$

де m – розмірність вектора входів.

2.2. Завдання на лабораторну роботу

1. Вибрати архітектуру та сформувати штучну нейронну мережу для вирішення задачі класифікації:

Варіант 1: автомобілів за двома ознаками (кількість пасажирських місць та максимальна вантажопідйомність) на два класи:

1) легкові – кількість пасажирів не більше 4 та вантажопідйомність не перевищує 500 кг;
2) вантажні – кількість пасажирів більше 4 та вантажопідйомність понад 500 кг;

Варіант 2: атлетів за двома ознаками (зріст та вага) на два класи:

1) жокеї – зріст не більше 170 см та вага не перевищує 65 кг;
2) баскетболісти – зріст більше 170 см та вага понад 65 кг;

Варіант 3: літературні твори за двома ознаками (кількість головних героїв та кількість сторінок) на два класи:

1) повісті – кількість головних героїв не більше 5 та кількість сторінок не перевищує 200;
2) романи – кількість головних героїв більше 5 та кількість сторінок понад 200.

Для вирішення задачі необхідно:

- сформувати навчальну вибірку, яка містить достатню кількість шаблонів, кожен з яких включає вхідний вектор та відповідний вихідний вектор;

- сформувати персептрон відповідної розмірності, вказавши в якості функції навчання '**learnpn**';
- реалізувати процес навчання персептрона за допомогою функції **adapt** або **train;**
- шляхом моделювання перевірити результат роботи навченої мережі.

2. Побудувати графік, який зображує простір входів та лінії, що відображають, як налаштована нейронна мережа ділить простір входів на підпростори, які відповідають різним класам.

3. Вибрати архітектуру та сформувати штучну нейронну мережу для вирішення задачі класифікації двоелементних вхідних векторів відповідно до рис. 2.1. Масштаб графіка вибрати самостійно.

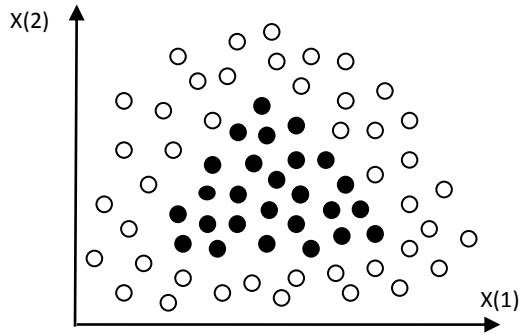


Рисунок 2.1

4. Перевірити працездатність навченої мережі за допомогою моделювання.

Контрольні питання

1. В яких випадках для навчання персептрона доцільно застосовувати нормовану функцію навчання?

2. Чи впливає на результат навчання персептрона послідовність пред'явлення навчальних шаблонів?

3. Чи можна добитися розділення вхідних образів відповідно до рис. 2.1 за допомогою персептрона, що містить нейрони з нульовими значеннями зміщень?

ЛАБОРАТОРНА РОБОТА 3

Апроксимація лінійної функції за допомогою адаптивної лінійної нейронної мережі

3.1 Інформаційний матеріал

Для вирішення задач апроксимації функції, фільтрації сигналів та прогнозування використовуються адаптивні лінійні нейронні мережі, які подібно до персептронів здатні вирішувати лише лінійно розв'язні задачі. Однак для їх навчання використовується більш потужна процедура Уідроу-Хоффа [4], заснована на методі градієнтного спуску. До складу адаптивних лінійних нейронних мереж зазвичай входить лінія затримки, що надає їм можливість опрацьовувати послідовності вхідних даних, тобто часові ряди.

Для формування одношарової лінійної мережі в пакеті програм Neural Network Toolbox використовується функція **linearlayer**:

> net = linearlayer (inputDelays, WidrowHoffLR).

Вхідні аргументи:

inputDelays – опис лінії затримки на вході мережі в наступному форматі: min_delay : max_delay (за замовчуванням 1:2);

WidrowHoffLR – величина швидкості навчання в правилах Уідроу-Хоффа (за замовчуванням 0,01).

Вихідні аргументи:

net – об'єкт класу Neural Network з архітектурою лінійного адаптивного шару нейронів.

Приклад. Сформувати адаптивну лінійну нейронну мережу, яка містить лінію затримки з трьох блоків, величина швидкості навчання дорівнює 0,001:

> net = linearlayer (1:3, 0.001).

Щоб гарантувати збіжність процесу навчання, швидкість навчання повинна бути менше норми найбільшого власного вектора матриці кореляції векторів входу. Кількість нейронів у шарі визначається автоматично в процесі навчання на підставі розмірності вихідних даних.

Підготовка даних для навчання здійснюється за допомогою функції **preparets**, яка автоматично зміщує вхідний та цільовий часові ряди на стільки кроків, скільки необхідно для заповнення початкових станів блоків лінії затримки:

> [Xs, Xi, Ai, Ts] = preparets (net, X, T).

Вхідні аргументи:

net – ім'я нейронної мережі до навчання;

X – послідовність незміщених вхідних даних;

T – послідовність незміщених цільових (вихідних) даних.

Вихідні аргументи:

Xs – послідовність зміщених вхідних даних, з яких вилучено дані для встановлення початкових станів блоків лінії затримки;

Xi – послідовність початкових станів блоків лінії затримки;

Ai – послідовність початкових станів при наявності зворотного зв'язку (за замовчуванням – порожній масив);

Ts – послідовність зміщених вихідних даних мережі, з яких вилучено дані в кількості, яка відповідає кількості блоків лінії затримки.

Приклад. Підготувати дані для навчання адаптивної лінійної мережі, яка містить лінію затримки з двох блоків:

> X = {–1 0 1 2 3};

> T = { 0 1 2 3 4};

> net = linearlayer (1:2, 0.01);

> [Xs, Xi, Ai, Ts] = preparets (net, X, T).

Результат:

> Xs = {1 2 3}

> Xi = {–1 0}

> Ai = Empty cell array: 1-by-0

> Ts = {2 3 4}

Кожен раз, коли змінюється кількість блоків лінії затримки, необхідно використовувати функцію **preparets** для відповідного переформатування вхідних та цільових даних.

Для навчання адаптивних лінійних нейронних мереж використовується функція **train**. Оскільки величина швидкості навчання є дуже малою, для досягнення бажаних результатів необхідно встановити кількість епох навчання достатньо великою.

Приклад. Установити кількість епох навчання рівною 1000, порогове значення помилки 0,001 та здійснити навчання мережі з попереднього прикладу:

> net.trainParam.epochs = 1000;

> net.trainParam.goal = 0.001;

> net_train = train (net, Xs, Ts, Xi, Ai).

Якщо навчання мережі виконано із застосуванням даних, підготовлених за допомогою функції **preparets**, то при моделюванні необхідно окремо вказувати послідовність вхідних даних та початкові стани на виході лінії затримки:

> Y = sim (net_train, Xs, Xi).

Адаптивна лінійна нейронна мережа відноситься до класу динамічних, тому допускається лише *послідовний* спосіб представлення вхідних даних у вигляді масиву типу **cell array**. Для підготовки навчальних даних використовується функція **con2seq**, яка перетворює числовий масив розміру

$n \times m$, що відповідає груповому представленню даних, в масив комірок розміру $1 \times m$, що містить числові масиви розміру $n \times 1$ та відповідає послідовному представленню даних.

Приклад. Перетворити числовий масив P формату **double array** розміру 2×3 в масив X формату **cell array** розміру 1×3, який містить комірки розміру 2×1:

> P = [1 3 5; 2 4 6];

> X = con2seq (P).

Результат: X = $\left\{\begin{bmatrix}1\\2\end{bmatrix};\begin{bmatrix}3\\4\end{bmatrix};\begin{bmatrix}5\\6\end{bmatrix}\right\}$.

Приклад. Дискретизувати безперервну лінійну функцію $y(t) = \sin(4\pi t)$ на інтервалі 2 секунди з частотою 40 вимірювань в секунду та перетворити в масив комірок X:

> time = 0 : 0.025 : 2;

> P = sin (4*pi*time);

> X = con2seq (P).

Для зворотного перетворення послідовного масиву комірок в звичайну матрицю використовується функція **cat.**

Приклад. Побудувати графік, на якому кружками червоного кольору відобразити значення масиву комірок X з попереднього прикладу:

> X1 = cat (2, X{:});

> plot (time, X1, 'or').

3.2. Завдання на лабораторну роботу

Експеримент 1.

1. Сформувати адаптивну лінійну нейронну мережу з одним входом та одним виходом, лінія затримки складається з двох блоків.

2. Підготувати дані та виконати навчання мережі так, щоб у відповідь на вхідний сигнал у вигляді послідовності значень {5 4 3 2} на виході мережі формувалася послідовність значень {10 20 30 40}.

3. Дослідити вплив кількості блоків, з яких складається лінія затримки, на точність роботи мережі. Експериментально підібрати оптимальну кількість блоків затримки, при якій помилка на виході мережі досягає мінімального значення.

Експеримент 2.

1. Сформувати функцію $x(t) = \sin(2\pi t)$ тривалістю 2.5 сек, та виконати її дискретизацію з частотою 50 вимірювань в секунду.

2. Сформувати адаптивну лінійну нейронну мережу з одним входом та одним виходом, лінія затримки складається з двох блоків.

3. Підготувати дані та виконати навчання мережі так, щоб у відповідь на послідовність вхідних сигналів, які відповідають функції $x(t)$, на виході мережі формувалася послідовність значень, що відповідають перетворенню $y(t) = 2x(t) + 3$.

4. Побудувати графік, що відображає значення помилки на виході мережі.

5. Дослідити вплив кількості блоків, з яких складається лінія затримки, на точність роботи мережі. Експериментально підібрати оптимальну кількість блоків затримки, при якій помилка на виході мережі досягає мінімального значення.

Експеримент 3.

1. Сформувати функцію $x(t) = \sin(2\pi t)$ та виконати її дискретизацію так, щоб перші 3 сек вимірювання виконувалися з частотою 100 разів на секунду, а наступні 2 сек частота функції збільшилася вдвічі $\omega = 4\pi$, а вимірювання проводилися 40 разів на секунду.

2. Сформувати адаптивну лінійну нейронну мережу з одним входом та одним виходом, лінія затримки складається з двох блоків.

3. Підготувати дані та виконати навчання мережі так, щоб у відповідь на послідовність вхідних сигналів, які відповідають функції $x(t)$, на виході мережі формувалася затримана на один крок послідовність значень, що відповідають перетворенню $y(t) = 3x(t-1) + 1$.

4. Побудувати графік, що відображає значення помилки на виході мережі.

5. Дослідити вплив кількості блоків, з яких складається лінія затримки, на точність роботи мережі. Експериментально підібрати оптимальну кількість блоків затримки, при якій помилка на виході мережі досягає мінімального значення.

6. Зробити висновки щодо впливу кількості блоків в лінії затримки на точність роботи адаптивної лінійної нейронної мережі.

Контрольні питання

1. Чому при виборі занадто великого значення швидкості навчання величина помилки на виході мережі починає зростати?

2. Як треба формувати опис лінії затримки на вході алаптивної лінійної мережі для вирішення задачі прогнозування?

3. Чим відрізняється реакція адаптивної лінійної мережі на подачу на вхід масиву [1 2 3 4] та послідовності значень {1 2 3 4}?

ЛАБОРАТОРНА РОБОТА 4

Класифікація образів за допомогою багатошарових нейронних мереж прямого поширення сигналів

4.1. Інформаційний матеріал

Для вирішення завдань, які не можуть бути вирішені за допомогою лінійних нейронних мереж, використовуються багатошарові мережі прямого поширення сигналів. Мережа з одним прихованим шаром здатна здійснити будь-яке нелінійне перетворення «вхід-вихід». Від умов конкретної задачі залежить кількість нейронів у вихідному шарі мережі, проте кількість прихованих шарів та їх розмір визначаються на основі досвіду або інтуїції розробника. Навчання мережі зазвичай включає кілька кроків:

- вибір початкової архітектури мережі з використанням, наприклад, такого евристичного правила: кількість нейронів прихованого шару визначається половиною сумарної кількості входів та виходів мережі;
- ітеративний процес навчання мережі з застосуванням однієї з модифікацій методу зворотного поширення помилки;
- якщо бажану якість роботи мережі не було досягнуто, слід збільшити число нейронів прихованого шару або збільшити кількість шарів.

Для створення багатошарової нейронної мережі прямого поширення сигналів, навчання якої виконується за методом зворотного поширення помилки, в пакеті програм Neural Network Toolbox використовується функція **feedforwardnet**:

> net = feedforwardnet (hiddenSizes, trainFcn).

Вхідні аргументи:

hiddenSizes – вектор-рядок, який описує розмірність прихованих шарів мережі;

trainFcn – функція навчання, що реалізує метод зворотного поширення (за замовчуванням 'trainlm').

Вихідні аргументи:

net – об’єкт класу Neural Network з архітектурою багатошарової мережі.

Як функція навчання може бути обрана одна з модифікацій методу зворотного поширення: '**traingd**' – стандартний алгоритм градієнтного спуску, '**traingda**' – градієнтний спуск з вибором величини швидкості навчання, '**traingdm**' – градієнтний спуск з інерцією, '**traingdx**' – градієнтний спуск зі змінною швидкістю навчання, '**trainbr**' – метод зворотного поширення з Байєсовою регуляризацією, '**trainrp**' – пороговий алгоритм зворотного поширення, '**trainscg**' – масштабований метод сполученого градієнта, '**traincgf**' – метод сполученого градієнта зі зворотним поширенням помилки в модифікації Флетчера-Ривса, '**traincgp**' – метод сполученого

градієнта в модифікації Полака-Рібейри, **'traincgb'** – метод сполученого градієнта з перезапусками Пауелла-Біла, **'trainbfg'** – квазіньютонов алгоритм в модифікації Бройдена, Флетчера, Гольдфарба та Шанно, **'trainoss'** – однокроковий алгоритм методу січної, **'trainlm'** – алгоритм Левенберга-Марквардта.

Приклад. Сформувати нейронну мережу прямого поширення сигналів, у якій перший прихований шар містить 3 нейрона, другий – 2 нейрона; для навчання використовується пороговий алгоритм зворотного поширення:

> net = feedforwardnet ([3 2], 'trainrp').

Кожна функція навчання характеризується набором параметрів, значення яких визначені за замовчуванням. Наприклад, **'trainlm'** має наступні параметри:

```
> net.trainParam
ans =
    Function Parameters for 'trainlm'
    show: 25                – кратність виведення інформації на графік,
                              виміряна в кількості епох навчання;
  epochs: 1000              – максимальна кількість циклів навчання;
    time: Inf               – граничний час навчання;
    goal: 0                 – цільове значення помилки на виході мережі;
min_grad: 1e–07             – мінімальне допустиме значення градієнта;
max_fail: 6                 – максимальна кількість перевірок, при яких
                              спрацьовує «ранній останов»;
      mu: 0.001             – початкове значення параметра μ;
  mu_dec: 0.1               – коефіцієнт приросту параметра μ;
  mu_inc: 10                – коефіцієнт зменшення параметра μ;
 mu_max: 10000000000        – максимальне допустиме значення μ.
```

При необхідності значення будь-якого параметра може бути змінено.

Приклад. Встановити кількість циклів навчання 300, цільове значення помилки на виході 0,0001, інтервал виведення інформації на графік 50:

```
> net.trainParam.epochs = 300;
> net.trainParam.goal = 1e–4;
> net.trainParam.show = 50.
```

Функція **'trainlm'** обрана в якості навчальної за замовчуванням, оскільки забезпечує максимальну швидкодію, але вимагає значних ресурсів пам’яті. Якщо ресурси недостатні, тобто термін обчислення однієї ітерації дуже великий, слід вибрати квазіньютонов алгоритм або одну з модифікацій методу сполученого градієнта. Пороговий алгоритм градієнтного спуску

також пред'являє мінімальні вимоги до обсягу оперативної пам'яті та характеризуэться достатньою швидкодією.

В пакеті MATLAB передбачені функції для підготовки даних, що використовуються при моделюванні. Для формування навчальних шаблонів з метою навчання нейронної мережі розпізнаванню символів латинського алфавіту використовується функція **prprob**:

> [alphabet, targets] = prprob.

В результаті генеруються масиви **alphabet** та **targets**, які використовуються при навчанні мережі в якості вхідних даних та цільових виходів. Масив **alphabet** визначає 26 векторів входу, кожен з яких описує відповідну букву алфавіту у вигляді шаблону розміром 7×5 та представлений вектором-стовпцем з 35 бінарних елементів. Масив **targets** визначає 26 векторів виходу, кожен з яких містить 26 елементів, один з них (номер якого збігається з номером букви в алфавіті) дорівнює 1, а інші дорівнюють 0. Для графічного відображення символу використовується функція **plotchar**.

Приклад. Подати графічно 7-й символ латинського алфавіту:

> [alphabet, targets] = prprob;

> alphabet (: , 7)

> plotchar (alphabet (: , 7)).

Результат представлено на рис. 4.1.

alphabet(: , 7) = 0 1 1 1 0
1 0 0 0 1
1 0 0 0 0
1 0 0 0 0
1 0 0 1 1
1 0 0 0 1
0 1 1 1 0

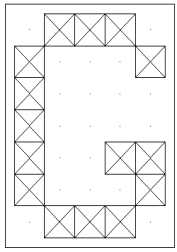


Рисунок 4.1

4.2. Завдання на лабораторну роботу

1. Вибрати архітектуру та сформувати нейронну мережу прямого поширення сигналів для реалізації логічної операції XOR («виключає АБО»), таблиця істинності якої виглядає наступним чином:

X(1)	X(2)	Y
0	0	0
1	0	1
0	1	1
1	1	0

2. Підготувати дані та виконати навчання мережі, використовуючи одну з модифікацій методу зворотного поширення помилки.

3. Протестувати навчену нейронну мережу.

4. Вибрати архітектуру та сформувати нейронну мережу прямого поширення сигналів для вирішення задачі розпізнавання символів латинського алфавіту. При подачі на вхід бінарного вектора, що представляє букву, мережа повинна формувати на виході вектор, в якому елемент, що відповідає порядковому номеру букви в алфавіті, дорівнює одиниці, а інші елементи дорівнюють нулю.

5. Підготувати дані та виконати навчання нейронної мережі.

6. Дослідити результати роботи навченої мережі в умовах дії шуму. Шум моделюється за допомогою функції **randn** у вигляді вектора-стовпця з 35 елементів, значення яких є випадковими величинами з інтервалу [–1 1] із середнім значенням 0 та стандартним відхиленням, яке є меншим або дорівнює 0,5. Вектор шуму додається до відповідного вектора-стовпця масиву **alphabet**.

7. Побудувати графік, який відображає залежність величини помилки на виході мережі від рівня шуму. Помилка дорівнює евклідовій нормі різниці вектора виходу мережі та відповідного цільового вектора. Для побудови графіка необхідно:

1) вектори, що моделюють шум, формуються із середнім значенням 0 та стандартним відхиленням від 0 до 0,5 з кроком 0,05;

2) для кожного рівня шуму формується 10 зашумлених вхідних векторів для кожного символу алфавіту;

3) моделюється вихід мережі у відповідь на кожен зашумлений вхідний вектор;

4) обчислюється середня сумарна величина помилки на виході мережі для кожного рівня шуму від 0 до 0,5, яка записується в масив;

5) отримані масиви значень рівня шуму та величини помилки мережі використовуються для побудови графіка.

Контрольні питання

1. Які активаційні функції можливо використовувати для нейронів, що застосовуються в нейронних мережах прямого поширення сигналів, навчання яких виконується за методом зворотного поширення помилки?

2. Які критерії завершення процесу навчання нейронної мережі за методом зворотного поширення помилки?

3. Назвіть переваги та недоліки модифікацій алгоритму зворотного поширення помилки, які реалізовані в пакеті Neural Network Toolbox.

ЛАБОРАТОРНА РОБОТА 5

Апроксимація функцій та класифікація образів за допомогою радіальних базисних нейронних мереж

5.1. Інформаційний матеріал

Нейронні мережі, в яких використовуються нейрони з радіальними базисними активаційними функціями, є альтернативою багатошарових нейронних мереж прямого поширення сигналів.

Для вирішення задачі апроксимації функції за допомогою двошарової радіальної базисної мережі в пакеті програм Neural Network Toolbox використовується функція **newrb**, яка ітеративно збільшує кількість нейронів в прихованому шарі мережі до тих пір, поки на виході не буде досягнуто задане значення середньої квадратичної помилки:

> net = newrb (X, T, goal, spread, MN, DF).

Вхідні аргументи:

X – масив розмірністю R×Q, який містить Q вхідних векторів, кожен з яких складається з R елементів;

T – масив розмірністю S×Q, який містить Q цільових векторів, кожен з яких складається з S елементів;

goal – цільове значення середньої квадратичної помилки на виході мережі (за замовчуванням 0);

spread – ширина (радіус) радіальної базисної функції (за замовчуванням 1,0);

MN – максимальна кількість нейронів радіального базисного шару (за замовчуванням дорівнює Q);

DF – кратність виведення інформації на графік, виміряна в кількості доданих нейронів (за замовчуванням дорівнює 25).

Вихідні аргументи:

net – об’єкт класу Neural Network з архітектурою радіальної базисної мережі.

В процесі побудови мережі на екран виводиться графік, що відображає залежність помилки на виході мережі від кількості нейронів в радіальному шарі. Той же результат можна отримати швидше за допомогою функції **newrbe**, але без побудови графіка та виведення інформації на екран:

> net = newrbe (X, T, spread).

Для вирішення задачі регресії використовується функція **newgrnn**, яка формує узагальнену регресійну нейронну мережу та має аналогічний синтаксис:

> net = newgrnn (X, T, spread).

Якість апроксимації залежить від значення параметра **spread**. Його слід вибирати більшим, ніж крок розбиття інтервалу завдання навчальної послідовності даних, але меншим розміру інтервалу. …

Приклад. Сформувати радіальну базисну нейронну мережу для апроксимації функції, яка відображає значення аргументу [1 2 3] у відповідні значення функції [2,0 4,1 5,9] так, щоб значення середньої квадратичної помилки не перевищувало 0,1:

> X = [1 2 3];

> T = [2.0 4.1 5.9];

> net = newrbe (X, T, 0.1).

Для визначення кількості нейронів з радіальною базисною активаційною функцією, яку містить перший шар мережі, використовується команда:

> net.layers{1}.size.

Імовірнісні нейронні мережі [4] – це різновид радіальних базисних мереж, що застосовується для вирішення задач класифікації образів. Для формування ймовірнісної мережі використовується функція **newpnn**:

> net = newpnn (X, T, spread),

проте, в даному випадку, на відміну від розглянутих вище, значення параметра **spread** за замовчуванням дорівнює 0,1. В результаті формується двошарова нейронна мережа, архітектура якої збігається з архітектурою радіальної базисної мережі, але в вихідному шарі застосовується «конкурентна» активаційна функція, яка вибирає нейрон з максимальним значенням виходу та встановлює його рівним 1, а решту обнуляє.

Приклад. Задача класифікації задана у вигляді масиву входів X та відповідного масиву індексів класів Tc:

> X = [1 2 3 4 5 6 7];

> Tc = [1 2 3 2 2 3 1];

тобто, вхідні значення “1” та “7” відносяться до класу «1»; значення “2”, “4” та “5” – до класу «2»; значення “3” та “6” – до класу «3». Необхідно перетворити індекси класів в масив цільових векторів та сформувати імовірнісну нейронну мережу для вирішення задачі класифікації:

> T = ind2vec (Tc);

> net = newpnn (X, T).

Функція **ind2vec** перетворює вектор індексів класів в матрицю зв’язності з одиницею в кожному стовпці, розташованою в рядку, що відповідає значенню індексу класу. У наведеному прикладі масив цільових векторів (матриця зв’язності) матиме вигляд:

T = [1 0 0 0 0 0 1;
0 1 0 1 1 0 0;
0 0 1 0 0 1 0].

5.2. Завдання на лабораторну роботу

1. Сформувати радіальну базисну нейронну мережу для апроксимації функції двох змінних, які приймають значення в межах:

$x_1 = -3 : 0.5 : 3;$

$x_2 = -2 : 0.5 : 2;$

а функція задана формулою $f(x_1, x_2) = 2\exp(2 - 5x_1^2) + 5(x_1 - x_2^2).$

2. Дослідити залежність кількості нейронів в радіальному базисному шарі, які знадобилося створити для досягнення заданої точності, від величини помилки на виході мережі, значення якої наведено в масиві [0,1 0,01 0,001].

3. Сформувати масив вхідних значень для тестування мережі, зменшивши в 2 рази крок дискретизації: $x_1 = -3 : 0.25 : 3;$

$x_2 = -2 : 0.25 : 2.$

4. Обчислити значення апроксимованої функції, одержувані на виході нейронної мережі при подачі на вхід масиву тестових значень.

5. Обчислити масив значень помилок апроксимації як різницю між значеннями функції, отриманими за допомогою формули, та результатами, отриманими на виході нейронної мережі.

6. За допомогою функції **surf** побудувати графічне зображення апроксимованої функції, отриманої в результаті моделювання.

7. Побудувати графік залежності значення помилки апроксимації від значень вхідних аргументів.

8. За допомогою функції **prprob** підготувати вхідні дані та сформувати імовірнісну нейронну мережу для класифікації символів латинського алфавіту.

9. Дослідити результати роботи імовірнісної мережі в умовах дії шуму, який моделюється випадковими величинами з інтервалу [–1 1] із середнім значенням 0 та стандартним відхиленням від 0 до 0,5 з кроком 0,05.

Контрольні питання

1. Які активаційні функції використовуються для нейронів, що застосовуються в радіальних базисних та імовірнісних нейронних мережах?

2. Чи може радіальна базисна нейронна мережа реалізувати логічну операцію XOR («виключає АБО»)? Обгрунтуйте свою відповідь.

3. Чому дорівнює максимальна кількість нейронів радіального базисного шару регресійної нейронної мережі?

4. Чому дорівнює кількість нейронів прихованого шару імовірнісної нейронної мережі?

ЛАБОРАТОРНА РОБОТА 6

Кластеризація даних за допомогою нейронних мереж Кохонена, що самоорганізуються

6.1. Інформаційний матеріал

У процесі аналізу великих масивів даних виникають задачі, пов'язані з дослідженням топологічної структури даних, їх об'єднанням в групи (кластери), яким притаманні деякі загальні властивості. Такі задачі можуть бути вирішені із застосуванням штучних нейронних мереж, що самоорганізуються. При цьому слід розрізняти мережі, що самоорганізуються, які містять нейрони, не зв'язані між собою, та називаються *шарами Кохонена*, і мережі з упорядкованими нейронами, які називають *картами Кохонена*. Тейво Кохонен – це відомий фінський вчений, який зробив великий внесок у вивчення штучних нейронних мереж.

Для формування шару Кохонена в пакеті програм Neural Network Toolbox використовується функція **competlayer**:

> net = competlayer (numClasses, kohonenLR, conscienceLR).

Вхідні аргументи:

numClasses – кількість нейронів у шарі, яка дорівнює кількості кластерів вхідних даних (за замовчуванням 5);

kohonenLR – величина швидкості навчання, яка застосовується для налаштування ваг (за замовчуванням 0,01);

conscienceLR – величина швидкості навчання, яка застосовується для налаштування зміщень (за замовчуванням 0,001);

Вихідні аргументи:

net – об'єкт класу Neural Network з архітектурою шара Кохонена.

Приклад. Сформувати шар Кохонена для вирішення задачі кластеризації даних, якщо кількість кластерів дорівнює трьом:

> net = competlayer (3).

В результаті буде сформовано шар Кохонена, у якого початкові значення вектора-рядка ваг нейронів не визначені, а вектор-стовпець зміщень нейронів містить невеликі додатні значення.

Оскільки в мережах, що самоорганізуються, реалізована процедура «навчання без учителя», то в якості параметрів функції навчання вказуються лише ім'я мережі та масив вхідних значень:

> net = train (net, inputs).

Конвертація результатів кластеризації, отриманих за допомогою мережі Кохонена у вигляді вектора-стовпця, що містить одиницю в тому розряді, який визначає номер кластера, в індекс кластера виконується за допомогою функції **vec2ind**:

> outputs = sim (net, inputs);

> classes = vec2ind (outputs).

Для формування карти Кохонена використовується функція **selforgmap**:

> net = selforgmap (dimens, coverSteps, initNeighbor, topologyFcn, distanceFcn).

Вхідні аргументи:

dimens – вектор-рядок, що визначає кількість нейронів по кожній розмірності карти (за замовчуванням – двовимірна карта з числом нейронів [8 8]);

coverSteps – кількість кроків першої фази навчання (за замовчуванням 100);

initNeighbor – початковий розмір ансамблю нейронів, які входять до найближчих сусідів (за замовчуванням 3);

topologyFcn – функція топології карти (за замовчуванням **'hextop'**);

distanceFcn – функція обчислення відстані між нейронами (за замовчуванням **'linkdist'**);

Вихідні аргументи:

net – об'єкт класу Neural Network з архітектурою багатовимірної карти Кохонена, що саморганізується.

Топологія карти Кохонена може бути визначена за допомогою однієї з трьох функцій: **'gridtop'** – прямокутна, **'hextop'** – гексагональна (шестикутна) та **'randtop'** – випадкова. Для обчислення відстані між нейронами використовуються функції: **'dist'** – обчислює масив евклідових відстаней між рядками матриці ваг та стовпцями матриці входів; **'linkdist'** – обчислює масив відстаней між нейронами шару, якщо заданий масив ваг нейронів, які трактуються як координати в просторі входів; **'boxdist'** – обчислює масив відстаней максимального координатного зсуву між нейронами шару, якщо заданий масив ваг нейронів (використовується в мережах з функцією топології **'gridtop'**); **'mandist'** – обчислює «відстань Мангеттена», тобто масив відстаней у вигляді сумарного координатного зсуву між кожним рядком матриці ваг та стовпцем матриці входів.

Приклад. Сформувати двовимірну карту Кохонена з прямокутною топологією та числом нейронів 3×5:

> net = selforgmap ([3 5], 100, 3, 'gridtop').

Після завершення навчання карти Кохонена користувач має можливість, натискаючи відповідну кнопку на панелі навчання, відкрити в новому графічному вікні наступні результати:

1) «SOM Topology» (англ.: self-organizing map, SOM) – графічне зображення обраного типу топології карти, де кожен шестикутник зображує нейрон.

2) «SOM Neighbor Connections» – сині шестикутники зображують нейрони, а червоні лінії відображають наявність зв'язків між сусідніми нейронами.

3) «SOM Neighbor Distances» – зображення матриці вагової відстані: кольори в областях, що містять червоні лінії, позначають відстані між нейронами: більш темні кольори представляють великі відстані; світліші кольори – менші відстані. Наприклад, на рис. 6.1 смуга темних сегментів перетинається від області нижнього центру до верхньої правої області, тобто мережа Кохонена розділила вхідні дані на два кластера.

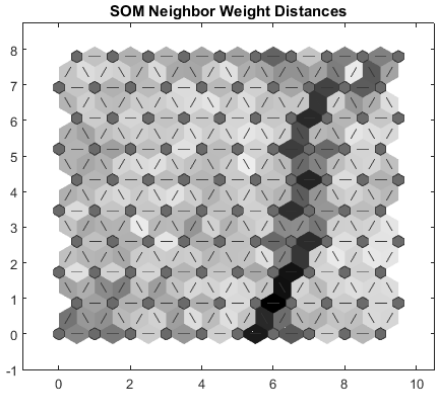


Рисунок 6.1

4) «SOM Weight Planes» – зображення площини ваг (також їх називають площинами компонентів) для кожного елемента вхідного вектора. На рис. 6.2 наведено приклад, в якому розмірність вхідного вектора дорівнює 2. Результат є візуалізацією ваг, які з'єднують кожен вхід з кожним нейроном. Темніші кольори відповідають великим значенням ваг. У наведеному прикладі ваги, які з'єднують нейрони зі входом 1, сильно відрізняються від ваг, що з'єднують нейрони зі входом 2.

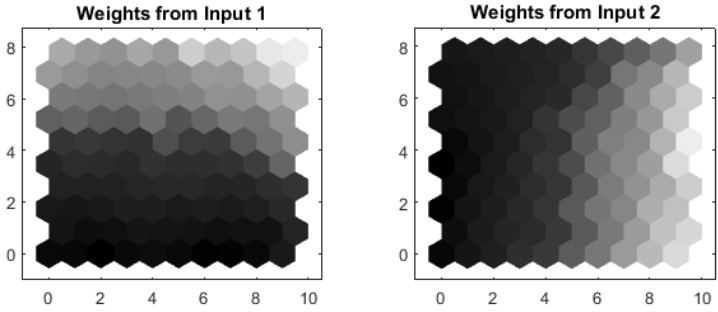


Рисунок 6.2

5) «SOM Sample Hits» – зображення розташування нейронів у топології із зазначенням, скільки разів кожен нейрон ставав «переможцем» в процесі навчання. У прикладі, наведеному на рис. 6.3, максимальна кількість «перемог», пов'язаних з будь-яким нейроном, становить 31.

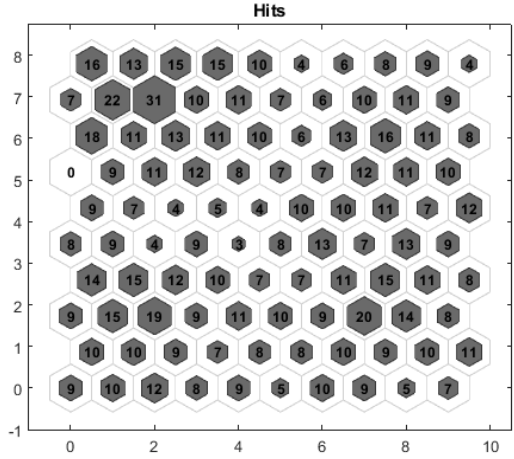


Рисунок 6.3

6) «SOM Weight Positions» – зображення вхідних даних зеленими кружками у просторі входів, а також розташування нейронів карти та взаємозв'язків між ними після завершення навчання (див. рис. 6.4).

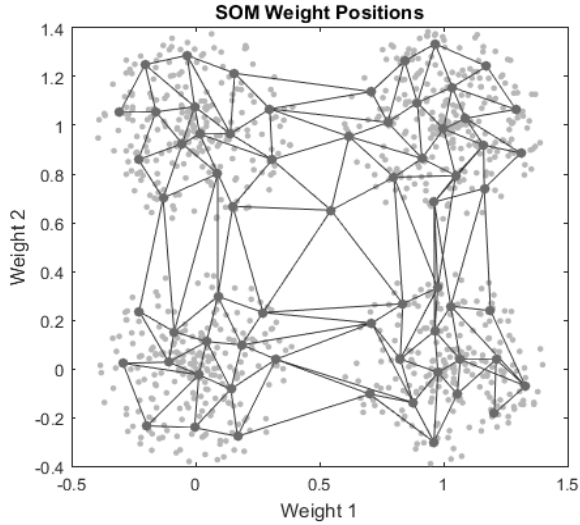


Рисунок 6.4

Щоб отримати аналогічні результати в режимі командної строки використовуються, відповідно, функції:

> plotsomtop (net);

> plotsomnc (net);

> plotsomnd (net);

> plotsomplanes (net);

> plotsomhits (net, inputs);

> plotsompos (net, inputs).

Приклад. Відобразити на екрані розташування нейронів дво- або тривимірної карти Кохонена:

> plotsompos (net); % 1 варіант

> plotsom (net.IW{1,1}, net.layers{1}.distances); % 2 варіант.

Підготовка даних для задач кластеризації даних виконується за допомогою функції **nngenc**:

> X = nngenc (PR, clusters, points, std_dev).

Вхідні аргументи:

PR – масив розміру R×2 мінімальних та максимальних значень центрів кластерів, де R – розмірність векторів входу;

clusters – кількість кластерів;

points – кількість об'єктів в кожному кластері;

std_dev – значення середньоквадратичного відхилення координат об'єктів від центру кластера.

Вихідні аргументи:

X – масив, що містить вхідні дані, розподілені на кластери.

Приклад. Сформувати та вивести на екран 5 кластерів, що містять по 8 двоелементних векторів, значення яких лежать в діапазоні [0 1], величина середньоквадратичного відхилення дорівнює 0,05:

> PR = [0 1; 0 1];

> clusters = 5;

> points = 8;

> std_dev = 0.05;

> X = nngenc (PR, clusters, points, std_dev);

> plot (X(1, :), X(2, :), '+').

Результат представлено на рис. 6.5.

6.2. Завдання на лабораторну роботу

1. Підготувати масив даних, який містить 48 двоелементних векторів, що утворюють 6 кластерів. Діапазон значень та середньоквадратичне відхилення вибрати самостійно. Відобразити на графіку розташування векторів.

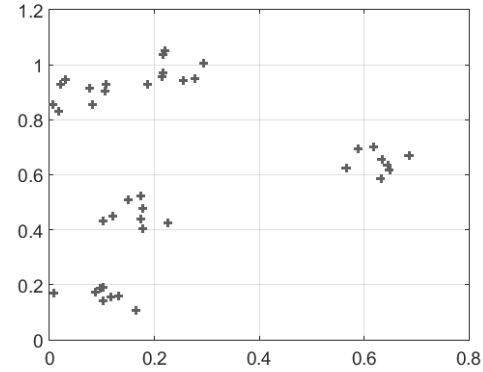


Рисунок 6.5

2. Сформувати шар Кохонена, що самоорганізується, для вирішення задачі кластеризації даних.

3. Виконати навчання нейронної мережі, створивши цикл для відображення на графіку розташування нейронів у просторі входів через кожні 5 епох навчання.

4. За допомогою моделювання перевірити результат вирішення задачі кластеризації, подаючи на вхід мережі вектори, що не використовувалися в процесі навчання, та конвертувати вихід мережі в індекс кластера.

5. Шляхом моделювання підібрати найкращий розмір та тип топології двовимірної карти Кохонена для вирішення задачі кластеризації на підставі сформованих раніше даних.

6. Підготувати масив даних, який містить 24 трьохелементних вектора, що утворюють 4 кластера: діапазон значень центрів кластерів [0 10], середньоквадратичне відхилення дорівнює 0,8. Відобразити на графіку розташування векторів.

7. Шляхом моделювання підібрати найкращий розмір та тип топології двовимірної карти Кохонена для вирішення задачі кластеризації даних.

Контрольні питання

1. Яка активаційна функція використовується для нейронів, які застосовуються в нейронних мережах, що самоорганізуються?

2. Для чого необхідно обчислювати відстань між нейронами, які складають карту Кохонена?

3. Поясніть зміст терміну «мертві нейрони».

4. Чи будуть результати навчання нейронної мережі Кохонена, що самоорганізується, тотожні, якщо процес навчання повторити кілька разів?

ЛАБОРАТОРНА РОБОТА 7

Класифікація даних за допомогою нейронних мереж векторного квантування

7.1. Інформаційний матеріал

Нейронні мережі векторного квантування або LVQ-мережі (англ.: Learning Vector Quantization) є розвитком мереж Кохонена, що самоорганізуються, і використовуються для класифікації вхідних векторів, які не є лінійно роздільними. Векторне квантування є більш загальною задачею, ніж кластеризація, оскільки передбачається, що кластери розділені між собою, тоді як сукупності для різних кодових векторів не обов'язково представляють собою роздільні кластери.

Нейромережі LVQ складаються з двох шарів. Перший шар відображає вхідні вектори в кластери, які знаходить мережа в процесі навчання. Другий рівень об'єднує групи кластерів першого рівня в класи, визначені цільовими даними. Для створення LVQ-мереж в пакеті Neural Network Toolbox використовується функція **lvqnet**:

> net = lvqnet (hiddenSize, lvqLR, lvqLF).

Вхідні аргументи:

hiddenSize – кількість нейронів прихованого шару (за замовчуванням 10);

lvqLR – величина швидкості навчання (за замовчуванням 0,01);

lvqLF – функція, яка застосовується для навчання мережі (за замовчуванням 'learnlv1').

Вихідні аргументи:

net – об'єкт класу Neural Network з архітектурою LVQ-мережі.

Приклад. Задана навчальна вибірка у вигляді масиву X, що містить десять двовимірних вхідних векторів, та масиву Tc, що містить індекси класів, до яких належать вхідні вектори:

```
> X =  [–3 –2 –2 –1  1 –1  1  2   2   3;
         0  1 –1 –2 –2  2  2  1 –1  0];
> Tc =  [ 1  1   1   2   2  2  2  1   1  1].
```

Необхідно створити LVQ-мережу для класифікації даних.

Для визначення кількості кластерів та, відповідно, кількості нейронів прихованого шару необхідно графічно представити розташування вхідних векторів. При цьому доцільно скористатися функцією **find**, яка повертає індекси елементів масиву, для яких виконується задана умова:

```
> I1 = find (Tc == 1);
> I2 = find (Tc == 2).
```

Зобразимо на графіку вектори, що належать різним класам, різними маркерами та різними кольорами:

> figure(1);

> axis([–4, 4, –3, 3]); % функція, яка визначає граничні значення осей графіка

> hold on; % функція, яка вказує, що наступні графічні результати виводитимуться в поточне графічне вікно

> plot (X(1, I1), X(2, I1), '+g');

> plot (X(1, I2), X(2, I2), 'ob').

Результат представлено на рис. 7.1.

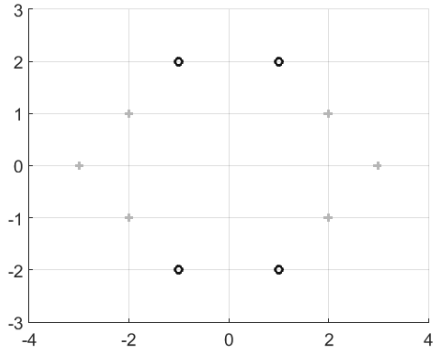


Рисунок 7.1

Аналіз отриманого зображення дозволяє зробити висновок, що вхідні вектори утворюють чотири кластери, що належать двом класам. Тоді формування LVQ-мережі виконується таким чином:

> net = lvqnet (4).

Для реалізації процедури навчання LVQ-мережі необхідно перетворити масив індексів класів Tc в масив цільових векторів. Для цього використовується функція **ind2vec**, яка описана в розділі 5.1, а переглянути результат дозволяє функція **full**:

> T = ind2vec (Tc);

> full (T)

ans = 1 1 1 0 0 0 0 1 1 1

0 0 0 1 1 1 1 0 0 0

Для навчання мережі застосовується функція **train**:

> net_train = train (net, X, T).

Після навчання ваги нейронів конкурентного шару визначають положення центрів кластерів. Щоб перевірити якість функціонування навченої мережі, на її вхід подається масив вхідних векторів та результат перетворюється в масив індексів класів за допомогою функції **vec2ind**:

> Y = sim (net_train, X);

> Yc = vec2ind (Y).

7.2. Завдання на лабораторну роботу

1. Сформувати мережу векторного квантування для реалізації логічної операції XOR, таблиця істинності якої наведена в розділі 4.2.

2. Графічно відобразити вхідні вектори так, щоб вектори, які належать різним класам, відображалися різними маркерами.

3. Виконати навчання мережі та відобразити на тому ж графіку положення центрів кластерів, яким відповідають нейрони прихованого шару.

4. За допомогою моделювання перевірити ефективність роботи мережі.

5. Сформувати мережу векторного квантування для класифікації тривимірних образів на два класи. Варіанти навчальних вибірок представлено в таблицях:

Варіант 1

Образ	Клас
[1; 1; –1]	1
[1; 0; –1]	1
[1; –1; 1]	1
[–1; 0; 1]	1
[0; 0; 0]	2
[0; 0; –1]	2
[1; –1; 1]	2
[1; –1; 0]	2
[–1; 1; –1]	2
[–1; 0; –1]	2
[–2; 2; –2]	1
[–2; 1; –2]	1

Варіант 2

Образ	Клас
[–3; 3; –3]	1
[0; 3; –3]	1
[3; –3; 3]	1
[0; –3; 3]	1
[–3; 3; 3]	2
[0; 3; 3]	2
[3; 3; –3]	2
[3; 3; 0]	2
[3; –3; –3]	2
[0; –3; –3]	2
[4; –4; 4]	2
[4; –3; 4]	2

6. Графічно відобразити вхідні вектори так, щоб вектори, які належать різним класам, відображалися різними маркерами.

7. Виконати навчання мережі спочатку за допомогою правила LVQ1 (функція навчання 'learnlv1'), а потім – правила LVQ2 (функція 'learnlv2').

8. За допомогою моделювання порівняти ефективність роботи двох варіантів мережі, подаючи на вхід вектори, які не використовувалися при навчанні.

Контрольні питання

1. У чому полягають відмінності правила навчання LVQ2 від правила навчання LVQ1?

2. Чи може в LVQ-мережах кількість нейронів лінійного шару перевищувати кількість нейронів конкуруючого шару? Обгрунтуйте свою відповідь.

ЛАБОРАТОРНА РОБОТА 8

Моделювання часових рядів за допомогою рекурентних нейронних мереж

8.1. Інформаційний матеріал

Характерною особливістю архітектури рекурентних нейронних мереж є наявність зворотних зв'язків, що дозволяє обробляти послідовності даних та враховувати передісторію процесів, що спостерігаються. Найпростішим типом рекурентних мереж (англ.: Simple Recurrent Network, SRN) є мережі Елмана, що складаються з двох шарів, в яких прихований шар охоплений динамічним зворотним зв'язком.

Для створення мережі Елмана в пакеті програм Neural Network Toolbox використовується функція **layrecnet**:

> net = layrecnet (layerDelays, hiddenSizes, trainFcn).

Вхідні аргументи:

layerDelays – опис лінії затримки в ланцюзі зворотного зв'язку в наступному форматі: min_delay : max_delay (за замовчуванням 1:2);

hiddenSizes – вектор-рядок, який описує розмірність прихованих шарів мережі (за замовчуванням 10);

trainFcn – функція навчання, що реалізує метод зворотного поширення (за замовчуванням 'trainlm').

Вихідні аргументи:

net – об'єкт класу Neural Network з архітектурою рекурентної мережі. При цьому нейрони прихованого шару мають тангенціальну активаційну функцію 'tansig', а нейрони вихідного шару – лінійну 'purelin'.

Приклад. Створити мережу Елмана, прихований шар якої містить 5 нейронів, а сигнал зворотного зв'язку затримується на один крок:

>net = layrecnet (1, 5).

Для графічного відображення архітектури побудованої мережі використовується функція **view**(net). Результат представлено на рис. 8.1.

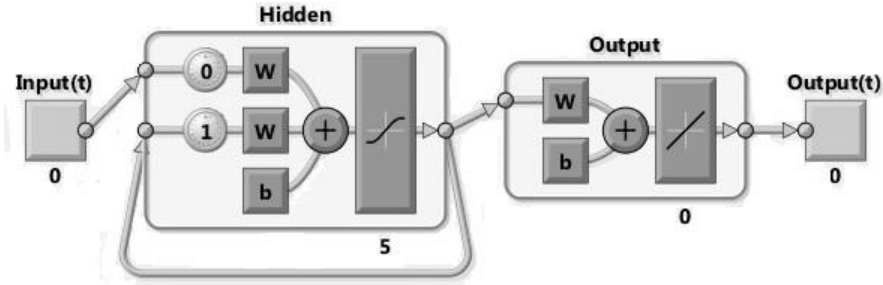


Рисунок 8.1

Розмірність вхідних даних на рис. 8.1 та розмір вихідного шару нейронів дорівнюють нулю. Тобто, ці параметри остаточно будуть визначені в процесі навчання нейронної мережі.

В якості функції навчання може бути обрана будь-яка, що реалізує одну з модифікацій методу зворотного поширення, які розглянуто в розділі 4.1.

Оскільки мережі Елмана відносяться до класу динамічних мереж, вхідні та цільові дані, які використовуються для навчання, повинні бути представлені в *послідовному* форматі **cell array**. Для перетворення групового представлення даних в послідовність використовується функція **con2seq**. Підготовка даних для навчання рекурентних мереж може здійснюватись за допомогою функції **preparets.** Опис вказаних функцій наведено в розділі 3.1.

8.2. Завдання на лабораторну роботу

1. Вирішити задачу розпізнавання у вхідній послідовності, що складається з нулів та одиниць, двох одиниць, які слідують підряд. Для цього необхідно:

- вибрати розмірність прихованого шару та створити мережу Елмана;
- сформувати вхідну послідовність з нулів та одиниць так, щоб в ній були присутні всі можливі комбінації;
- сформувати відповідну вихідну послідовність сигналів, елементи якої приймають значення «1» тільки в тому випадку, коли у вхідній послідовності поточне та попереднє значення дорівнюють одиниці;
- виконати навчання мережі;
- перевірити результат навчання за допомогою моделювання, подавши на вхід мережі послідовність нулів та одиниць, яка не використовувалася в процесі навчання;
- дослідити вплив кількості нейронів в прихованому шарі на швидкість навчання та якість роботи мережі.

2. Навчити мережу Елмана виконанню перетворення XOR («виключає АБО») в динаміці. Для цього необхідно:

- вибрати розмірність прихованого шару та створити мережу Елмана;
- сформувати вхідну послідовність з нулів та одиниць так, щоб в ній були присутні всі можливі комбінації;
- сформувати відповідну вихідну послідовність, значення елементів якої визначаються двома попередніми значеннями входу відповідно до таблиці істинності операції XOR, яку наведено в розділі 4.2;
- виконати навчання мережі;
- перевірити результат навчання за допомогою моделювання, подавши на вхід мережі послідовність нулів та одиниць, яка не використовувалася в процесі навчання.

Контрольні питання

1. У чому полягають переваги рекурентних мереж порівняно з іншими типами штучних нейронних мереж?
2. Чим відрізняється навчання мереж Елмана від навчання багатошарових мереж прямого поширення сигналів?

ЛАБОРАТОРНА РОБОТА 9
Моделювання асоціативної пам'яті за допомогою нейронної мережі Хопфілда

9.1. Інформаційний матеріал

Нейронна мережа Хопфілда є рекурентною мережею, в якій в процесі динаміки вихідні сигнали подаються знову на вхід. При подачі на вхід деякого вхідного вектора мережа за кінцеве число ітерацій переходить до стану стійкої рівноваги, що дозволяє асоціювати поданий вектор із одним з образів, які зберігаються в пам'яті. Таким чином, мережа Хопфілда володіє асоціативними можливостями для вилучення інформації, яка була запам'ятована. Обсяг асоціативної пам'яті мережі Хопфілда визначається кількістю нейронів, з яких вона складається.

У пакеті Neural Network Toolbox реалізовано модифікований метод синтезу мережі Хопфілда, який відрізняється від класичної моделі [1], але є більш простим для моделювання. Для створення мережі Хопфілда використовується функція **newhop**:

> net = newhop (T).

Вхідний аргумент – масив T розмірністю R×Q, який містить Q цільових векторів (зі значеннями 1 або –1), де R – ромірність вхідного вектора.

Вихідний аргумент – об'єкт класу Neural Network з архітектурою мережі Хопфілда.

Приклад. Створити мережу Хопфілда з двома стійкими точками [–1; 1; –1] та [1; –1; 1] в тривимірному просторі:

> T = [–1 1; 1 –1; –1 1];

> net = newhop (T).

Мережа Хопфілда складається з одного шару нейронів, який на вході має лінію затримки. Щоб виконати моделювання мережі необхідно вхідний вектор перетворити в *послідовний* формат та вказати його як початкові умови лінії затримки. Оскільки процес генерування вихідних сигналів мережі, які за зворотним зв'язком подаються на вхід, повторюється багато разів, то при

застосуванні функції **sim** в якості другого аргументу необхідно вказати послідовність з двох значень, перше – це такт дискретності, друге – кількість кроків моделювання, а в якості четвертого аргументу – початкові умови лінії затримки. Тоді результатом буде масив комірок, що містить послідовні значення виходів мережі, які формуються ітераційно. Для перегляду отриманих результатів дані необхідно перетворити у звичайний масив за допомогою функції **cat**.

Приклад. Виконати моделювання мережі Хопфілда протягом п’яти ітерацій, встановивши в якості початкових умов вектор-стовпець [–0,5; 0,6; –0,7]:

```
> T = [–0.5;  0.6;  –0.7];
> Tseq = con2seq (T);
> Yseq = sim (net, {1 5}, {}, Tseq);
> Y = cat (2, Yseq{:})
  Y =
      –0.6748   –0.8049   –0.9399   –1.0000   –1.0000
       0.6971    0.8099    0.9410    1.0000    1.0000
      –0.7194   –0.8149   –0.9421   –1.0000   –1.0000
```

Як видно з наведеного прикладу, із заданого початкового стану мережа перейшла в найбільш близьке стійке положення рівноваги. Бажано, щоб мережа поводилася аналогічним чином при завданні будь-якої початкової точки в межах гіперкуба, вершини якого складені з усіх комбінацій чисел 1 та –1 в просторі відповідної розмірності. Однак, одним з недоліків мереж Хопфілда є те, що результатом роботи мережі може стати генерація небажаного «паразитного» стану рівноваги, який не відповідає жодному з векторів, що були запам’ятовані.

9.2. Завдання на лабораторну роботу

1. За допомогою функції **prprob** сформувати масив **alphabet**, що містить 26 векторів, кожен з яких складається з 35 елементів (зі значеннями «1» або «0») та описує символ латинського алфавіту. Змінити отриманий масив таким чином, щоб він містив значення «1» та «–1».

2. Сформувати масив, що містить вектори, які описують перші п’ять символів латинського алфавіту.

3. Створити мережу Хопфілда з п’ятьма точками рівноваги, які відповідають першим п’яти символам.

4. Дослідити ефективність розпізнавання мережею Хопфілда перекручених символів, для чого виконати моделювання, вказавши в якості початкових умов вектор, що відповідає одному з символів, у якого змінені значення декількох елементів. Результат представити графічно за допомогою

функції **plotchar**, попередньо змінивши її для відображення результатів, що містять значення «1» та «–1» замість значень «1» та «0».

5. За допомогою функції **randn** сформувати вектор розмірністю 35×1, який містить випадкові значення з інтервалу [1 –1].

6. Виконати моделювання мережі Хопфілда, вказавши в якості початкових умов вектор, що містить випадкові значення.

7. Дослідити залежність ймовірності появи «паразитних» станів рівноваги від обсягу інформації, який мережа повинна запам'ятати: повторити пункти 2-6, сформувавши масив вхідних векторів, що містить більшу кількість символів латинського алфавіту.

Контрольні питання

1. Який тип асоціативної пам'яті можна реалізувати за допомогою нейронної мережі Хопфілда?

2. Обчислити матрицю ваг нейронної мережі Хопфілда, яка здатна запам'ятати та розпізнати вектори [1; –1] та [–1; –1].

3. Сформулювати достатню умову стійкості нейронної мережі Хопфілда.

Список літератури

1. Руденко О.Г., Бодянський Є.В. Штучні нейронні мережі: Навчальний посібник. – Київ : Компанія СМІТ, 2006. – 404 с.
2. Тимощук П.В. Штучні нейронні мережі: Навчальний посібник. – Львів : Видавництво Львівської політехніки, 2011. – 444 с.
3. Уосермен Ф. Нейрокомп'ютерна техніка: Теорія і практика / Пер. з англ. І.Ю. Юрчак, 2001. [Електронний ресурс] – URL: http://www.victoria.lviv.ua/html/wosserman/index.htm
4. Хайкин С. Нейронные сети: полный курс, 2-е издание / Пер. с англ. – М. : Изд-кий дом «Вильямс», 2006. – 1104 с.
5. Медведев В.С., Потемкин В.Г. Нейронные сети. MATLAB 6 / Под общ. ред. В.Г. Потемкина. – М. : ДИАЛОГ-МИФИ, 2002. – 496 с.
6. Оссовский С. Нейронные сети для обработки информации / Пер. с польского И.Д. Рудинского. – М. : Финансы и статистика, 2002. – 344 с.

Зміст

Навчальне видання

Методичні вказівки
до лабораторних робіт
з дисципліни «Штучні нейронні мережі»
для студентів спеціальностей 122 – Комп'ютерні науки
та 124 – Системний аналіз

Укладач ДОРОФЄЄВ Юрій Іванович

Відповідальний за випуск проф. Куценко О. С.

Роботу до видання рекомендував проф. Безменов М. І.

В авторській редакції

План 2019, поз. 293

Підп. до друку 26.11.19.	Формат 60х84 1/16.	Папір офсетний.
Друк – ризографія.	Гарнітура Таймс New Roman.	Ум. друк. арк. 0,8.
Обл.-вид. арк. 1,3.	Наклад 50 прим.	Зам. № 6358.
Ціна договірна.		

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2.

Надруковано у СПДФО Ковальчук Н.П.
Свідоцтво № 24800000000150925 від 01.08.2013 р.